Evolutionary Approach for Finding Correlation Immune Boolean Functions of Order t with Minimal Hamming Weight

Stjepan Picek[1], Sylvain Guilley[2], Claude Carlet[3], Domagoj Jakobovic[4], and Julian F. Miller[5]

[1] KU Leuven, ESAT/COSIC and iMinds
Kasteelpark Arenberg 10, bus 2452, B-3001 Leuven-Heverlee, Belgium
[2] TELECOM-ParisTech, Paris, France & Secure-IC S.A.S., Rennes, France
[3] LAGA, UMR 7539, CNRS, Department of Mathematics
University of Paris 8 and University of Paris 13
2 Rue de la Liberté, 93526 Saint-Denis Cedex, France
[4] Faculty of Electrical Engineering and Computing, University of Zagreb, Croatia
[5] Department of Electronics, University of York, York, UK

Abstract. The role of Boolean functions is prominent in several areas like cryptography, sequences and coding theory. Therefore, various methods to construct Boolean functions with desired properties are of direct interest. When concentrating on Boolean functions and their role in cryptography, we observe that new motivations and hence new properties have emerged during the years. It is important to note that there are still many design criteria left unexplored and this is where Evolutionary Computation can play a distinct role. One combination of design criteria that has appeared recently is finding Boolean functions that have various orders of correlation immunity and minimal Hamming weight. Surprisingly, most of the more traditionally used methods for Boolean function generation are inadequate in this domain. In this paper, we concentrate on a detailed exploration of several evolutionary algorithms and their applicability for this problem. Our results show that such algorithms are a viable choice when evolving Boolean functions with minimal Hamming weight and certain order of correlation immunity. This approach is also successful in obtaining Boolean functions with several values that were known previously to be theoretically optimal, but no one succeeded in finding actual Boolean functions with such values.

Keywords: Boolean Functions, Cryptography, Correlation Immunity, Hamming Weight, Evolutionary Algorithms

1 Introduction

One usual source (although not the only one) of nonlinearity in ciphers are Boolean functions. In block ciphers, the nonlinearity often comes from Substitution Boxes or S-boxes which are actually a number of Boolean functions (hence,

also the name vectorial Boolean functions). On the other hand, in stream ciphers the nonlinearity comes from Boolean functions. Both of those scenarios, while not the only ones, show us the prominent role of Boolean functions in cryptography. Finding Boolean functions fitting all the criteria and analyzing the best possible trade-offs between these criteria are still crucial questions today.

Historically, Boolean functions have been dominantly used in conjunction with Linear Feedback Shift Registers (LFSRs). Two commonly used models are filter generators and combiner generators. In a combiner generator, several LFSRs are used in parallel and their output is the input for a Boolean function. On the other hand, in a filter generator, the output is obtained by a nonlinear combination of a number of positions in a longer LFSR [3]. To be effective such Boolean functions need to be balanced, have high nonlinearity, large algebraic degree, large algebraic immunity, and high correlation immunity (in the case of combiner generators).

To obtain such functions, there exist a number of construction methods. Those methods can be roughly divided into algebraic constructions, random search, heuristics and combinations of those methods [19]. In this paper, we examine one branch of heuristics, more precisely Evolutionary Algorithms (EAs), in order to evolve Boolean functions. It is worth mentioning that EAs can be used either as the primary or the secondary construction method. In primary constructions one obtains new functions without using known ones. In secondary constructions, one uses already known Boolean functions to construct new ones (either with different properties or sizes) [3].

We said that a Boolean function needs to be balanced (among other criteria) to be suitable for cryptography. Indeed, this is true, but only when we consider the role of Boolean functions in filter and combiner generators. However, recently one more application emerged where we are actually interested in Boolean functions that have minimal Hamming weight and are therefore as far as possible from being balanced. Such Boolean functions can be used to help resist side-channel attacks.

Side-channel attacks do not rely on the security of the underlying algorithm, but rather on the implementation of the algorithm in a device [13]. One class of countermeasures against side-channel attacks are masking schemes. In masking schemes one randomizes the intermediate values that are processed by the cryptographic device. One obvious drawback of such an approach is the masking overhead which can be substantial in embedded devices or smart cards.

Correlation immune Boolean functions can reduce the masking overhead either by applying leakage squeezing method [4, 6] or with Rotating S-box masking [5]. We emphasize that a number of construction methods (primarily algebraic constructions) are not suitable for these design criteria since they produce balanced Boolean functions.

Up to now, there has been almost no work to examine how to evolve Boolean functions with various orders of correlation immunity and minimal Hamming weight. This is the gap this paper aims to rectify. In order to do so, we experiment with several algorithms, both from the single objective and the multi-objective

optimization area. More precisely, we use Genetic Algorithms (GAs), Genetic Programming (GP), Cartesian Genetic Programming (CGP), and NSGA-II. Our investigation has a twofold impact since we offer a detailed examination of the EAs performance on the aforementioned problem. Furthermore, we find values previously completely unknown for certain Boolean function sizes and orders of the correlation immunity property.

1.1 Related Work

There exist a number of works that examine Boolean functions in cryptography and their generation with Evolutionary Computation (EC) techniques. Here, we give only a small subset of works related to our investigation.

Millan et al. work with GAs in order to evolve Boolean functions with high nonlinearity [15]. Burnett in her thesis uses GAs to evolve both Boolean functions and Substitution boxes [2]. McLaughlin and Clark use simulated annealing to evolve Boolean functions that have several cryptographic properties with optimal values [14]. Picek, Jakobovic and Golub experiment with GP and GAs to find Boolean functions that have several optimal properties [18]. Picek et al. experiment with both heuristics and heuristics in conjunction with algebraic construction to evolve Boolean functions with high nonlinearity [20]. Picek et al. use CGP to evolve Boolean functions with eight inputs and high nonlinearity [19]. Finally, Picek et al. investigate several EAs in order to evolve Boolean functions with different values of the correlation immunity property. In the same paper, the authors also discuss the problem of finding correlation immune functions with minimal Hamming weight, but they experiment only with Boolean functions that have eight inputs [17].

The remainder of this paper is organized as follows. In Section 2, we describe relevant cryptographic properties and representations of Boolean functions. Section 3 represents the techniques for using Boolean functions in masking schemes as well as our motivation for this research. In Section 4, experimental setup and the algorithms we use are given. Section 5 presents the results and a short discussion. Finally, Section 6 concludes and gives some suggestions for future work.

2 Introduction to Boolean Functions and Their Properties

Let n, m be positive integers, i.e. $n, m \in \mathbb{N}^+$. The set of all n-tuples of the elements in the field $\mathbb{F}_2$ is denoted as $\mathbb{F}_2^n$ where $\mathbb{F}_2$ is the Galois field with 2 elements. The inner product of two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ is denoted as $\boldsymbol{a} \cdot \boldsymbol{b}$ and equals $\boldsymbol{a} \cdot \boldsymbol{b} = \oplus_{i=1}^{n} a_i b_i$. Here, "$\oplus$" represents addition modulo two (bitwise XOR). The Hamming weight (HW) of a vector $\boldsymbol{a}$, where $\boldsymbol{a} \in \mathbb{F}_2^n$, is the number of non-zero positions in the vector.

An (n, m)-function is any mapping F from $\mathbb{F}_2^n$ to $\mathbb{F}_2^m$. If m equals 1 then the function f is called a Boolean function.

A Boolean function f on $\mathbb{F}_2^n$ can be uniquely represented by a truth table (TT), which is a vector $(f(\mathbf{0}), ..., f(\mathbf{1}))$ that contains the function values of f, ordered lexicographically, i.e. $\boldsymbol{a} \leq \boldsymbol{b}$ [3].

The support $supp(a)$ of a vector a is the index set of the non-zero positions in a, i.e. $supp(a) = \{i : a_i \neq 0\}$, and the support $supp(f)$ of a Boolean function f is the vector set of the non-zero entries in the truth table (TT) representation of f, i.e. $supp(f) = \{x : f(x) \neq 0\}$ [3]. The HW of a Boolean function f is the cardinality of its support.

The Walsh-Hadamard transform W_f is a second unique representation of a Boolean function that measures the correlation between $f(\boldsymbol{x})$ and the linear function $\boldsymbol{a} \cdot \boldsymbol{x}$ [3]:

$$W_f(\boldsymbol{a}) = \sum_{\boldsymbol{x} \in \mathbb{F}_2^n} (-1)^{f(\boldsymbol{x}) \oplus \boldsymbol{a} \cdot \boldsymbol{x}}. \tag{1}$$

A Boolean function f is correlation immune of order t (in brief, $CI(t)$) if the output of the function is statistically independent of the combination of any t of its inputs [21]. For the Walsh-Hadamard spectrum it holds equivalently [10]:

$$W_f(\boldsymbol{a}) = 0, \text{ for } 1 \leq HW(\boldsymbol{a}) \leq t. \tag{2}$$

3 Boolean Functions and Masking

Some applications manipulate sensitive data, such as cryptographic keys. Obviously, these should remain secret. However, skillful attackers might try to probe bits within a processor or a memory; they often succeed, unless countermeasures are implemented [9].

To protect secrets from probing attempts, it is customary to implement a countermeasure known as masking. It consists in changing randomly the representation of the key (and of any other data which depends on the key), so as to deceive the attacker. For example, if each bit k_i, $1 \leq i \leq n$ of a key k is masked with a random bit m_i, then an attacker could probe $k_i \oplus m_i$. However, provided m_i is uniformly distributed, the knowledge of $k_i \oplus m_i$ does not disclose any information on bit k_i, unless of course the attacker can also probe separately m_i, in which case a higher order masking would be necessary.

However, for implementation reasons, it is often impractical to mask each bit individually. Indeed, the generation of random numbers is costly, thus it is desirable to limit the number of random bits required. Furthermore, masking each bit of a vector of length n would correspond to choosing the global mask in the whole set of 2^n possible masks, which may be too costly.

Specifically, on the example of the AES cipher, some key bytes are mixed with plaintext bytes before entering a Substitution box. Generating all the 256 Substitution boxes suitable for all possible masks in $\mathbb{F}_2^8$ is too expensive for embedded systems. Hence, by restricting the number of possible masks, the overhead incurred by the countermeasure becomes more affordable.

Let us consider the simple example of masking one byte ($n = 8$). This can be achieved by using two complementary masks, such as $m_0 = (00000000)_2$ and $m_1 = (11111111)_2$. An attacker who measures one bit of the masked byte cannot derive any information about the corresponding unmasked bit. If we define by f the Boolean function $\mathbb{F}_2^8 \to \mathbb{F}_2$ whose support is $\{m_0, m_1\}$, then f plays its masking role as it is balanced: any bit can take value 0 and 1 with equal probability. In general, any Boolean function which is $CI(1)$ is a valid masking.

Let us now consider a stronger attacker who is able to probe two bits simultaneously. In this case, some information can be recovered. Typically, if the two masked bits are equal, then so are the two unmasked bits. So, by testing all pairs of bits, the attacker can recover the whole key (or its complement). It happens that, against such "second-order attacker", it would be desirable that the masks be the support of a $CI(2)$ Boolean function [6]. But clearly, this support must have a cardinality strictly greater than 2. Hence, masking can be summarized as the problem of finding Boolean functions whose support is a set of masks with the two following constraints:

1. it should have small Hamming weight, for implementation reasons, and
2. it should have high correlation immunity t, in a view to resist an attacker with multiple ($\leq t$) probes.

Clearly, there is a tradeoff. This motivates the research for low Hamming weight high correlation immunity Boolean functions.

Some work on this topic has been summarized by Hedayat, Sloane and Stufken in their book [11]. However, in this book, some entries (minimum Hamming weight for a given pair (n, t)) are expressed as non-tight bounds. Recently, the exact value for entries corresponding to $(n = 9, t = 4)$, and $(n = 10, t \in \{4, 5\})$ have been obtained by Carlet and Guilley in [6]. Still, the exact values $(n, t) \in \{(11, 4-5), (12, 4-6), (13, 4-7)\}$ that are of practical interest, have remained unknown until this research.

4 Experimental Setup

In this section, we briefly present the algorithms we use as well as the experimental setup and fitness functions.

4.1 Single Objective Optimization

Genetic Algorithm. The GA represents the individuals as strings of bits representing truth tables of Boolean functions. We use a simple GA with elimination tournament selection with size 3 [8]. A mutation is selected uniformly at random between a simple mutation, where a single bit is inverted, and a mixed mutation, which randomly shuffles the bits in a randomly selected subset. The crossover operators are one-point and uniform crossover, performed uniformly at random for each new offspring. For each of the fitness functions we experiment with population sizes of 50, 100, 500, and 1 000 and mutation probabilities of 0.1, 0.3, 0.5, 0.7, and 0.9.

Genetic Programming. GP uses a representation where individuals are trees of Boolean primitives which are then evaluated according to the truth table they produce. The function set for GP in all experiments is OR, XOR, AND, XNOR, and AND with one input inverted. Terminals correspond to n Boolean variables. Boolean functions may be represented with only XOR and AND operators, but it is quite easy to transform it from one notation to the other. GP uses a tournament selection with tournament size 3 [12]. We use a simple tree crossover with 90% bias for functional nodes and a subtree mutation. We experiment with tree depth sizes of 5, 7, 8, and 9 and population sizes of 100, 200, 500, 1 000, and 2 000.

Cartesian Genetic Programming. The function set n_f for the CGP is the same as for the GP. Setting the number of rows to be 1 and levels-back parameter to be equal to the number of columns is regarded as the best and most general choice [16]. We experiment with genotype sizes of 500, 1 000, 2 000, and 3 000 and mutation rates of 1%, 4%, 7%, 10%, and 13%. The number of input connections n_n for each node is two and the number of program output connections n_o is one. The population size for CGP equals five in all our experiments. For CGP individual selection we use a (1 + 4)-ES in which offspring are favored over parents when they have a fitness better than or equal to the fitness of the parent. The mutation operator is one-point mutation where the mutation point is chosen with a fixed probability. The number of genes mutated is defined as fixed percentage of the total number of genes. CGP solutions are directed graphs with Boolean primitives as nodes, that are also evaluated using the truth table they produce.

Fitness Function. The following fitness function for single objective optimization is obtained after a set of experiments where we determined which one performed best on average. The goal is **maximization**:

$$fitness = (MAX\_HW - supp) - MAX\_HW \times |CI - TARGET\_CI|. \quad (3)$$

Here, MAX_HW represents the Hamming weight of a Boolean function that has all ones in its truth table (i.e. $HW = 2^n$, where n represents the number of inputs of a Boolean function), $TARGET\_CI$ represents the order of correlation immunity we want to find and finally, *supp* represents the cardinality of the support of a Boolean function. The function consists of two parts: the first part rewards Boolean functions with smaller support, while the second part acts as a penalty for solutions with a correlation immunity that differs from the target. The penalty part is multiplied with maximum value of the reward part, so that any solution with the right CI is always better than any other solution with different CI. This way, the distance to the target CI is regarded as a constraint, and the support as a secondary objective.

4.2 Multi-Objective Optimization

Since the goal of the function design includes two criteria, it can be formulated as a multi-objective optimization problem. The first objective is attaining a desired target correlation immunity, and the second one is the minimization of the support. Following these criteria, a multi-objective problem can be formulated as:

$$fitness_A = |CI - TARGET\_CI|; \tag{4}$$

$$fitness_B = MAX\_HW - supp, \tag{5}$$

where the first criteria, $fitness_A$, is **minimized**, while the second criteria, $fitness_B$, is **maximized**.

In our experiments we applied the well known NSGA-II algorithm for multi-objective optimization [7]. Note that NSGA-II can be paired with any of the Boolean representations (i.e. truth table in GA, tree in GP and graph in CGP), but based on the performance in the initial round of experiments, we only present the results of tree representation (GP) with multi-objective evolution.

4.3 Common Parameters

The number of independent runs for each experiment is 50. The function set n_f for both GP and CGP in all the experiments is OR, XOR, AND, XNOR, and AND with one input inverted. For stopping condition we use the number of evaluations which we set to 1 000 000.

5 Results and Discussion

We report the performance of the selected algorithms, and additionally present the best obtained values in order to compare EC with the existing results. For the first part, the evolutionary algorithms are compared with each other using basic statistical indicators to assess their performance. The comparison between EAs is carried out using the parameter combinations that fill the gaps in the recent work [6], i.e. for functions with the number of bits n and correlation immunity t in the set $(n, t) \in \{(11, 4-5), (12, 4-6), (13, 4-7)\}$.

For the second part, we only select the single best results obtained by any algorithm and compare it with the values found in the related literature. Since there is a large number of experiments, we conduct a parameter tuning phase for a medium sized Boolean function of nine inputs and the correlation immunity order of two. Parameter tuning phase has a stopping condition of 500 000 evaluations. Later we use the best obtained set of parameters for all test scenarios. Due to the lack of space, we do not present exhaustively the tuning phase results.

5.1 Genetic Algorithm

The results for GA in this application were very poor; in the tuning phase, where we test different parameters for Boolean functions of 9 bits and target correlation

immunity equal to the value of two, we were unable to obtain a single solution with the desired correlation immunity for any of the parameter settings. The GA with the truth table representation does succeed in finding the desired values, but only for very small problem sizes (e.g. for up to six variables), where the size of the solution is not large. However, since those cases are not representative to the problem, we do not experiment with GA in the rest of the paper.

5.2 Genetic Programming

Based on the results of the parameter tuning phase, the best performance for GP was obtained with a maximum tree depth of five, whereas there were practically no differences with regards to the population size. Based on this, we continued with depth five and the population size of 1 000.

However, since the tuning was performed on 9 bit functions, we note that for larger sizes (e.g. 12 or 13 bits), the maximum depth of five may simply be not enough to represent the desired behavior. Indeed, while with the depth of seven and the same number of evaluations we obtain *statistically* worse solutions in general, there were cases where a single best solution was reached with a depth of 7. This is further explored in Section 5.5, while for the algorithm comparison we remain with the depth of 5 where the results are given in Table 1.

The results for each combination of n and t are reported in the form of $avg/max/\#hits$, where avg represents the average best fitness value over 50 runs, max is the single best fitness value, and $\#hits$ is the number of runs in which the best value was reached. We chose this simple statistic because the fitness often assumes negative values (due to the penalty part in the fitness function) which are not very indicative, and since the observed algorithms often reach the same maximum value.

Table 1. Results for GP (single objective, maximization)

n \ t	4	5	6	7
11	1 830.4/1 920/15	1 643.52/1 792/21		
12	3 840/3 840/50	3 507.2/3840/5	2 928.64/3 072/43	
13	7 680/7 680/50	7 383.04/7 680/37	6021.12/6144/47	5 324.8/6 144/30

5.3 Cartesian Genetic Programming

In the case of CGP, the best parameters after the tuning phase were mutation probability of 13% and the genotype size of 500. The results of CGP with the obtained parameter settings are given in Table 2, with the same nomenclature as in Table 1. Negative value means that on average most of the runs did not succeed in finding any solutions with the $TARGET\_CI$ value. This suggests that the number of evaluations was too low for those problem instances.

Table 2. Results for CGP (single objective, maximization)

n \ t	4	5	6	7
11	1218.56/1 792/4	901.12/1 536/6		
12	2 549.76/3 584/9	1 515.52/3 072/6	-532.48/2 048/28	
13	4 587.52/7 168/2	1 638.4/6 144/8	-819.2/4 096/20	-3 604.48/4 096/3

5.4 Multi-objective Genetic Programming

In the last phase, we used NSGA-II multi-objective algorithm to try to reach the desired output while regarding both correlation immunity and Hamming weight as independent objectives. Although the name implies a genetic algorithm, in this work the multi-objective approach is used with tree-based representation of GP, since it exhibited the best performance.

The results for the multi-objective approach were disappointing; in most cases, the algorithm was unable to reach the desired target CI value, while the secondary objective was very bad even in the other cases. We found solutions only in cases when $n = 12$ and $CI = 4$ where the support equals 2 048 and when $n = 13$ and $CI = 4$ with the support equal to 4 096. In the first case the value was found three times, and in the second, only once.

5.5 Discussion

Finally, we combine all the single best values we obtained with any evolutionary algorithm and present them in Table 3. In this table the values are not represented with our fitness function, in which the expression $(MAX\_HW - supp)$ was maximized, but rather only as the resulting support ($supp$), since this form was used in the existing work (note that in this case the smaller values are the better ones). These best EC results are equal to those presented in Tables 1 and 2, with the exception of 13 bits and correlation immunity 6, where GP with depth 7 obtained a better result.

This paper reports results for Boolean functions which were previously unknown. These are indicated using gray cells in Table 3. When discussing the optimality of those results, we follow the conjectures from [1].

Here, $w_{n,t}$ represents the lowest weight of $CI(t)$ nonzero function of n variables. The conjecture was made in [1, Sec. C.2] that the values in each column of Table 3 are non-decreasing. The values for $(n, t) \in \{(11, 4-5)\}$ in Table 3 are interesting from this viewpoint: if the conjecture is true then they are optimal since they cannot be smaller than for $(n, t) \in \{(10, 4-5)\}$, but the conjecture may be false; further investigations are needed to clarify this point. If $(n, t) \in \{(11, 4-5)\}$ represent the minimal possible values; then since it is known from [1, Sec. C.1] that $w_{n,t} \geq 2w_{n-1,t-1}$, then the solution for $n = 12$ and $t \in \{5, 6\}$ has also a minimal Hamming weight. Finally, for $n = 13$, by following the same reasoning, Hamming weights for $t \in \{6, 7\}$ are again the optimal values, if the Hamming weight for $n = 11$ and $t = 4, 5$ is optimal. Actually, the value $w_{13,6} = 1\,024$ was

already known (see Table 12.1 at page 319 in [11]), hence it does not appear as a gray cell in Table 3. However, there are cases in which we were unable to obtain the target correlation immunity value, which are denoted with an x. There are also instances in which none of the EAs we implemented was able to reach a previously known optimal value, and those are marked in italic. We note that some of those optimal values are actually trivial to obtain and by slight adjustments in the fitness functions we could reach those levels. However, we decided to follow a 'black-box' principle where we do not use specifics about the problem. It is obvious that for some combinations of the number of bits and correlation immunity, the optimization problem as formulated in this work, becomes very hard. In these cases the fitness function is unable to lead the population in the desired direction, which merits further research on both the design of the fitness function and properties of the underlying fitness landscape, as well as the use of different representations and genetic operators. Besides the comparisons based solely on the obtained results, it is also possible to consider the speed of the procedure. Indeed, in [1], authors report that with the SMT solver, the resolution of the problem can last several days. Obtaining the optimal values for the largest Boolean function size of 13 inputs, with our approach lasted on average 30 minutes.

The failure of a GA with a bit-string representation to reach any meaningful results may be explained with two causes: the first one relates to the problem size, and its rapid increase with the number of bits. While the size of the truth table grows exponentially with the number of bits, the size of the search space grows with an even larger rate, which quickly renders a standard GA unusable. The second reason could be a high epistasis of the problem since the bits in the truth table are not independent. This in the conjunction with the fact that the correlation immunity property is concerned with the statistical independence of the bits on the output/input, could lead to a reason why higher values of correlation immunity present difficulty for GA in bitstring representation. This phenomenon can be also observed on the results from [17].

The multi-objective approach has also failed to produce competitive results. Although the nature of the problem suggests different criteria may be optimized independently, the NSGA-II explores a large region in the search space that is of no interest to the final goal. In this application, the stated objectives can clearly be regarded as a primary and a secondary one; first try to reach a desired correlation immunity, and then reduce the Hamming weight as much as possible. While the fitness function we presented may obviously be defined in many different ways, it is apparent that in the current form it is more effective than the multi-objective approach. There are undoubtedly more applications in cryptography where multi-objective optimization may prove useful, where a trade-off between different properties is sought by the system designer.

Table 3. Best obtained results.

n \ t	1	2	3	4	5	6	7	8	9	10	11	12	13
5	2	8	16	16	32								
6	2	8	16	32	32	64							
7	2	8	16	64	64	*128*	128						
8	2	*16*	16	64	128	128	*256*	256					
9	2	*16*	*32*	128	128	256	256	*512*	512				
10	2	*16*	*32*	128	256	512	512	x	*1 024*	1 024			
11	2	*16*	*32*	128	256	512	1 024	1 024	x	*2 048*	2 048		
12	2	16	*32*	*256*	256	1 024	1 024	x	x	x	*4 096*	4 096	
13	2	16	32	*256*	*512*	1 024	2 048	4 096	4 096	x	x	*8 192*	8 192

6 Conclusion and Future Work

In this paper we investigated the evolution of Boolean functions with minimal Hamming weight and various orders of the correlation immunity property. An approach based on Genetic Programming proved to be very successful since we obtained very good results for all Boolean function sizes from five to thirteen inputs. We emphasize that this approach also yielded previously unknown values, where for the most of those it is possible to show they represent global optima. In our future work, we plan to concentrate on even larger Boolean functions in an attempt to find more unknown values as well as the practical upper limit for the use of EAs for the evolution of Boolean functions.

References

1. Bhasin, S., Carlet, C., Guilley, S.: Theory of masking with codewords in hardware: low-weight dth-order correlation-immune boolean functions. Cryptology ePrint Archive, Report 2013/303 (2013), http://eprint.iacr.org/
2. Burnett, L.D.: Heuristic Optimization of Boolean Functions and Substitution Boxes for Cryptography. Ph.D. thesis, Queensland University of Technology (2005)
3. Carlet, C.: Boolean Functions for Cryptography and Error Correcting Codes. In: Crama, Y., Hammer, P.L. (eds.) Boolean Models and Methods in Mathematics, Computer Science, and Engineering, pp. 257–397. Cambridge University Press, New York, NY, USA, 1st edn. (2010)
4. Carlet, C., Danger, J.L., Guilley, S., Maghrebi, H.: Leakage Squeezing of Order Two. In: Galbraith, S., Nandi, M. (eds.) Progress in Cryptology - INDOCRYPT 2012, LNCS, vol. 7668, pp. 120–139. Springer Berlin Heidelberg (2012)
5. Carlet, C., Guilley, S.: Side-channel Indistinguishability. In: Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy. pp. 9:1–9:8. HASP '13, ACM, New York, NY, USA (2013)
6. Carlet, C., Guilley, S.: Correlation-immune Boolean functions for easing counter measures to side-channel attacks (chapter 3). In: Niederreiter, H., Ostafe, A., Panario, D., Winterhof, A. (eds.) Algebraic Curves and Finite Fields Cryptography and Other Applications, pp. 41–70. Radon Series on Computational and Applied Mathematics 16, De Gruyter (August 2014)
7. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6(2), 182–197 (Apr 2002)

8. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Springer-Verlag, Berlin Heidelberg New York, USA (2003)
9. Gammel, B.M., Mangard, S.: On the duality of probing and fault attacks. J. Electronic Testing 26(4), 483–493 (2010), http://dx.doi.org/10.1007/s10836-010-5160-0
10. Guo-Zhen, X., Massey, J.: A spectral characterization of correlation-immune combining functions. IEEE Transactions on Information Theory 34(3), 569–571 (May 1988)
11. Hedayat, A.S., Sloane, N.J.A., Stufken, J.: Orthogonal Arrays, Theory and Applications. Springer series in statistics, Springer, New York (1999), ISBN 978-0-387-98766-8
12. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge, MA, USA (1992)
13. Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks: Revealing the Secrets of Smart Cards (Advances in Information Security). Springer-Verlag New York, Inc., Secaucus, NJ, USA (2007)
14. McLaughlin, J., Clark, J.A.: Evolving balanced Boolean functions with optimal resistance to algebraic and fast algebraic attacks, maximal algebraic degree, and very high nonlinearity. Cryptology ePrint Archive, Report 2013/011 (2013), http://eprint.iacr.org/
15. Millan, W., Clark, A., Dawson, E.: Heuristic design of cryptographically strong balanced Boolean functions. In: Advances in Cryptology - EUROCRYPT '98. pp. 489–499 (1998)
16. Miller, J.F. (ed.): Cartesian Genetic Programming. Natural Computing Series, Springer Berlin Heidelberg (2011)
17. Picek, S., Carlet, C., Jakobovic, D., Miller, J.F., Batina, L.: Correlation immunity of boolean functions: An evolutionary algorithms perspective. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2015, Madrid, Spain, July 11-15, 2015. pp. 1095–1102 (2015)
18. Picek, S., Jakobovic, D., Golub, M.: Evolving Cryptographically Sound Boolean Functions. In: Proceedings of the 15th Annual Conference Companion on Genetic and Evolutionary Computation. pp. 191–192. GECCO '13 Companion, ACM, New York, NY, USA (2013)
19. Picek, S., Jakobovic, D., Miller, J.F., Marchiori, E., Batina, L.: Evolutionary methods for the construction of cryptographic boolean functions. In: Genetic Programming - 18th European Conference, EuroGP 2015, Copenhagen, Denmark, April 8-10, 2015, Proceedings. pp. 192–204 (2015)
20. Picek, S., Marchiori, E., Batina, L., Jakobovic, D.: Combining Evolutionary Computation and Algebraic Constructions to Find Cryptography-Relevant Boolean Functions. In: Parallel Problem Solving from Nature - PPSN XIII - 13th International Conference, Ljubljana, Slovenia, September 13-17, 2014. Proceedings. pp. 822–831 (2014)
21. Siegenthaler, T.: Correlation-immunity of Nonlinear Combining Functions for Cryptographic Applications (Corresp.). IEEE Transactions on Information Theory 30(5), 776–780 (Sep 2006)